Primal-Dual Active-Set Methods for Isotonic Regression and Trend Filtering

Zheng Han
Industrial and Systems Engineering
Lehigh University
Bethlehem, PA 18015
zhh210@lehigh.edu

Frank E. Curtis
Industrial and Systems Engineering
Lehigh University
Bethlehem, PA 18015
frank.e.curtis@gmail.com

Abstract

Isotonic regression (IR) is a non-parametric calibration method used in supervised learning. For performing large-scale IR, we propose a primal-dual active-set (PDAS) algorithm which, in contrast to the state-of-the-art Pool Adjacent Violators (PAV) algorithm, is easily warm-started and can be parallelized. This warm-starting capability makes it well-suited for online settings. We prove that, like the PAV algorithm, our PDAS algorithm for IR is convergent and has a work complexity of $\mathcal{O}(n)$, though our numerical experiments suggest that our PDAS algorithm is often faster than PAV. In addition, we propose PDAS variants (with safeguarding to ensure convergence) for solving related trend filtering (TF) problems, providing the results of experiments to illustrate their effectiveness.

1 Introduction

Isotonic regression is a non-parametric method for fitting an arbitrary monotone function to a dataset [1, 2] that has recently gained favor as a calibration method for supervised learning [3, 4, 5, 6]. A well-known and efficient method for solving IR problems is the Pool Adjacent Violators (PAV) algorithm [7]. This method is easily implemented and enjoys a convergence guarantee with a work complexity of $\mathcal{O}(n)$ where n is the dimension of the dataset. A drawback of the PAV algorithm in large-scale settings, however, is that it is inherently sequential. Consequently, in order to exploit parallelism, one has to resort to decomposing the IR problem [8], where deciding the number of processors is nontrivial. For example, a recent Spark implementation of a parallelized PAV method suffers from significant overhead [9]. In addition, since the PAV algorithm must be initialized from a particular starting point, it cannot be warm-started—a fact that is especially detrimental when a sequence of IR problems need to be solved [10], such as in online settings where data points are added continually. As an alternative, we propose a primal-dual active-set (PDAS) method for solving IR problems. Our PDAS algorithm also has a convergence guarantee for IR and a work complexity of $\mathcal{O}(n)$, but can be warm-started and is easily parallelized. We also provide PDAS algorithm variants for a related class of trend filtering (TF) problems. Alternative approaches for certain TF problems include interior point methods [11], specialized ADMM methods [12], and proximal methods [13], but advantages of active-set methods such as ours are that they may terminate finitely and often yield very accurate solutions. For reasons such as these, they may be favorable for certain applications such as switch point identification and time series segmentation. The results of numerical experiments are provided for all of our algorithm variants to illustrate their practical strengths.

This paper is organized as follows. In §2, we define and describe the IR and TF problems for which our algorithms are designed. In §3, we summarize and compare the well-known Pool Adjacent Violators (PAV) algorithm and our proposed primal-dual active-set (PDAS) method for solving IR problems. PDAS variants (with and without safeguards) for solving related TF problems are pre-

sented in §4. We report on our experimental results as well as our findings with other approaches in §5. Finally, concluding remarks are provided in §6.

2 Problem Descriptions

Isotonic Regression We consider the isotonic regression (IR) problem

$$\min_{\theta\in\mathbb{R}^n} \tfrac{1}{2}\sum_{i=1}^{n}\omega_i(y_i-\theta_i)^2 \;\text{ subject to }\; \theta_1\le\ldots\le\theta_n, \tag{IR}$$

where $y\in\mathbb{R}^n$ represents observed data and $\omega\in\mathbb{R}^n_+$ represents weights for the data fitting term. The goal of this optimization problem formulation is to determine a monotonically increasing step function that matches the observed data as closely as possible in a sense of distance defined by ω.

Trend Filtering Problem (IR) is related to a special case of the trend filtering problem

$$\min_{\theta\in\mathbb{R}^n}\ \phi(\theta),\ \text{ where }\ \phi(\theta)=f(\theta)+\lambda g(\theta), \tag{TF}$$

$f:\mathbb{R}^n\mapsto\mathbb{R}$ is smooth and convex, $\lambda>0$ is a regularization parameter, and the regularization function $g:\mathbb{R}^n\mapsto\mathbb{R}$ is convex but not necessarily smooth. In trend filtering or time series segmentation, f is usually chosen to measure the distance between θ and y while g imposes desired properties on the solution θ. A typical trend filtering problem has the form

$$f(\theta)=\tfrac{1}{2}\sum_{i=1}^{n}\omega_i(y_i-\theta_i)^2 \;\text{ with }\; g(\theta)=\|D\theta\|_1 \;\text{ or }\; g(\theta)=\|(D\theta)_+\|_1,$$

where D is a first-order (or higher) difference operator and $(\gamma)_+=\max\{\gamma,0\}$ (component-wise). Specifically, as described in [11], a k-th order difference matrix $D^{(k,n)}\in\mathbb{R}^{(n-k)\times n}$ is defined recursively via the relation $D^{(k,n)}=D^{(1,n-k+1)}D^{(k-1,n)}$. For example, the first and second order difference matrix $D^{(1,n)}$ and $D^{(2,n)}$ are defined, respectively, as

$$D^{(1,n)}=\begin{bmatrix}1&-1&&&\\&1&-1&&\\&&\ddots&\ddots&\\&&&1&-1\end{bmatrix}\ \text{and}\ D^{(2,n)}=\begin{bmatrix}1&-2&1&&&\\&1&-2&1&&\\&&\ddots&\ddots&\ddots&\\&&&1&-2&1\end{bmatrix}.$$

When $g(\theta)=\|(D^{(1,n)}\theta)_+\|_1$ and λ is sufficiently large, solving (TF) gives the solution of (IR).

3 Algorithms for Isotonic Regression

We describe and compare two efficient algorithms for solving problem (IR); in particular, the PAV and our proposed PDAS algorithms are described and their corresponding theoretical properties are summarized in §3.1 and §3.2, respectively. Throughout this and the subsequent sections, we borrow the following notation from [7] in the algorithm descriptions.

Notation Let J represent a partition of the variable indices $\{1,2,\ldots,n\}$ into ordered blocks $\{B_1,B_2,\ldots\}$ where each block consists of consecutive indices, i.e., each block has the form $\{p,\ldots,q\}$ for $p\le q$. The immediate predecessor (successor) of block B is denoted as B_- (B_+). By convention, B_- (B_+) equals $\emptyset$ when B is the initial (final) block. The weighted average of the elements of y in block B is denoted $\text{Av}(B)\ :=\ (\sum_{i=p}^{q}\omega_i y_i)/(\sum_{i=p}^{q}\omega_i)$. For each index $i\in B=\{p,\ldots,q\}$, we define the "lower" and "upper" sets

$$L_i(J)=\{p,p+1,\ldots,i\}\ \text{ and }\ U_i(J)=\{i+1,i+2,\ldots,q\}.$$

Hereinafter, we shall use L_i and U_i for brevity when their dependence on a particular J is clear.

3.1 PAV for Isotonic Regression

We describe briefly the PAV algorithm [7, 1, 14, 15, 2, 8], state its main theoretical properties, and discuss its important features in this section. To start with, we rephrase the PAV algorithm from [7], where the algorithm is shown to replicate a dual active-set method for quadratic optimization.

Algorithm 1 PAV for Isotonic Regression

Input the initial partition $J = \{\{1\}, \ldots, \{n\}\}$ and set $C = \{1\}$
loop
  if $C_+ = \emptyset$ **then**
    Terminate and **return** $\theta_i = \text{Av}(B)$ for each $i \in B$ for each $B \in J$
  if $\text{Av}(C) \leq \text{Av}(C_+)$ **then**
    Set $C \leftarrow C_+$
  else
    Set $J \leftarrow \big(J \backslash \{C, C_+\}\big) \cup (C \cup C_+)$ and $C \leftarrow C \cup C_+$ ▷ Merge C with C_+
    while $\text{Av}(C_-) > \text{Av}(C)$ and $C_- \neq \emptyset$ **do**
      Set $J \leftarrow \big(J \backslash \{C_-, C\}\big) \cup (C_- \cup C)$ and $C \leftarrow C_- \cup C$ ▷ Merge C with C_-

The main idea of Algorithm 1 can be understood as follows. Initially, each index is represented by a separate block. The algorithm then sequentially visits all blocks, merging a block with its successor whenever a "violator" is met, i.e., whenever a block has a weighted average greater than its successor. Once any merge occurs, the algorithm searches backwards to perform subsequent merges in order to ensure that, at the end of any **loop** iteration, no violators exist up to the furthest visited block. Once all blocks have been visited, no violator exists and the solution θ will be monotonically increasing.

An impressive property of Algorithm 1 is that by storing an intermediate value for each block and showing that at most n merge operations may occur, one obtains an efficient implementation that solves problem (IR) within $\mathcal{O}(n)$ elementary arithmetic operations [15]. Due to this fact and its good practical performance, Algorithm 1 has been popular since its invention. However, we argue that the PAV algorithm does have critical drawbacks when it comes to solving large-scale problems of interest today. First, Algorithm 1 must be initialized with $J = \{\{1\}, \ldots, \{n\}\}$, which is unfortunate when one has a better initial partition, such as when one is solving a sequence of related instances of (IR) [10]. In addition, the sequential nature of PAV makes it difficult to leverage multi-processor infrastructures.

3.2 PDAS for Isotonic Regression

Primal-dual active-set (PDAS) methods have been proposed in the literature for solving Linear Complementarity Problems (LCPs) [16], bound-constrained QPs (BQPs) [17], and generally-constrained QPs [18]. To our knowledge, however, the application and theoretical analysis of a tailored PDAS method for solving problem (IR) has not previously been studied.

In this section, we propose a PDAS method designed explicitly for (IR) and discuss its theoretical guarantees and practical benefits. We first reveal the relationship between problem (IR) and a special class of convex BQPs for which a primal-dual active-set (PDAS) method is known to be well-suited. We then propose our PDAS algorithm tailored for solving (IR). Finally, the complexity of our proposed algorithm is analyzed and its key features and properties are discussed.

(IR) **and** (BQP) Let $\Omega = \text{diag}(\omega_1, \ldots, \omega_n)$ be the diagonal weight matrix. The dual of (IR) is then

$$\min_{z \in \mathbb{R}^{n-1}_+} \tfrac{1}{2} z^T D \Omega^{-1} D^T z - y^T D^T z, \tag{BQP}$$

where

$$D\Omega^{-1}D^T = \begin{bmatrix} \frac{2}{\omega_1} & -\frac{1}{\omega_2} & 0 & \cdots & 0 \\ -\frac{1}{\omega_2} & \frac{2}{\omega_2} & -\frac{1}{\omega_3} & \cdots & 0 \\ 0 & -\frac{1}{\omega_3} & \frac{2}{\omega_3} & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & -\frac{1}{\omega_{n-2}} \\ 0 & 0 & \cdots & -\frac{1}{\omega_{n-2}} & \frac{2}{\omega_{n-1}} \end{bmatrix} \quad \text{and} \quad Dy = \begin{bmatrix} y_1 - y_2 \\ y_2 - y_3 \\ \vdots \\ y_{n-1} - y_n \end{bmatrix}.$$

Since $D\Omega^{-1}D^T$ is positive definite with non-positive off-diagonal entries, it is an M-matrix, meaning that (BQP) has a form for which a PDAS method is well-suited [17]. In descriptions of PDAS methods for BQP such as that in [17], the notion of a partition corresponds to a division of the index set for z into "active" and "inactive" sets. There is a one-to-one correspondence between a partition of (BQP) and that of (IR); specifically, the non-zero indices in z correspond to the boundaries of the blocks of a partition for problem (IR). We have the following result for the present setting.

Theorem 3.1 (Theorem 3.2, [17])**.** *If the PDAS method from [17] is applied to solve problem* (BQP)*, then the iterate sequence $\{z_k\}$ is nondecreasing, has $z_k \geq 0$ for all $k \geq 1$, and converges to the optimal solution of* (BQP)*.*

A PDAS Algorithm for (IR) One can apply the PDAS method from [17] to solve (IR) by applying the approach to its dual (BQP). However, a straightforward application would fail to exploit the special structure of problem (IR). Algorithm 2, on the other hand, generates the same sequence of iterates as the PDAS method of [17], but is written in a much more computationally efficient form.

Algorithm 2 PDAS for Isotonic Regression

1: Input an initial partition J_0
2: For each $i \in B = \{p, \ldots, q\} \in J_0$, set

$$\theta_i \leftarrow \frac{\sum_{i\in B}\omega_i y_i}{\sum_{i\in B}\omega_i} \text{ and } z_i \leftarrow \begin{cases} \omega_i(y_i - \theta_i) & \text{if } i = p \\ 0 & \text{if } i = q \neq n \\ z_{i-1} + \omega_i(y_i - \theta_i) & \text{otherwise} \end{cases}$$

3: Initialize $J_1 \leftarrow J_0$
4: **for each** $i \in B \in J_1$ with $z_i < 0$, set $J_1 \leftarrow (J_1 \backslash B) \cup \{L_i, U_i\}$ ▷ Split B into L_i and U_i
5: **for each** $B \in J_1$, set $\alpha_B \leftarrow \sum_{i\in B}\omega_i y_i$, $\beta_B \leftarrow \sum_{i\in B}\omega_i$, and $\mu_B \leftarrow \alpha_B/\beta_B$
6: **for** $k = 1, 2, \ldots$ **do**
7: Initialize $J_{k+1} \leftarrow J_k$
8: **for each** $\{B_s, \ldots, B_t\} \subseteq J_k$ with $\mu_{B_{s-1}} \leq \mu_{B_s} > \cdots > \mu_{B_t} \leq \mu_{B_{t+1}}$ ▷ Merge $\{B_s, \ldots, B_t\}$

$$\text{Let } N \leftarrow \bigcup_{j=s}^{t} B_j \text{ and update } J_{k+1} \leftarrow (J_{k+1} \cup N)\backslash\{B_s, \ldots, B_t\},$$

$$\text{Set } \alpha_N \leftarrow \sum_{j=s}^{t}\alpha_{B_j}, \ \beta_N \leftarrow \sum_{j=s}^{t}\beta_{B_j}, \text{ and } \mu_N \leftarrow \alpha_N/\beta_N$$

9: **if** $J_{k+1} = J_k$, **then** break
10: **for each** $B \in J_k$, set $\theta_i \leftarrow \mu_B$ for all $i \in B$

The major difference between Algorithms 1 and 2 is the manner in which blocks are merged. In Algorithm 1, each merge involves a single pair of adjacent blocks, whereas Algorithm 2 allows simultaneous merge operations, each of which may involve more than two blocks. In the following theorem, we prove that Algorithm 2, like the careful implementation [15] of Algorithm 1, solves problem (IR) in $\mathcal{O}(n)$ elementary arithmetic operations.

Theorem 3.2. *If Algorithm 2 is applied to solve problem* (IR)*, then it will yield the optimal solution for* (IR) *within $\mathcal{O}(n)$ elementary arithmetic operations.*

Proof. One can verify that Algorithm 2 is equivalent to applying the PDAS method from [17] to solve (BQP), from which it follows by Theorem 3.1 that it finds the optimal solution in finitely many iterations. The initialization process in Steps 1–5 requires $\mathcal{O}(n)$ elementary arithmetic operations as each step involves at most a constant number of calculations with each value from the dataset. As for the main loop involving Steps 6–9, the introduction of α and β ensures that the number of elementary arithmetic operations in merging two blocks becomes $\mathcal{O}(1)$. Thus, since the **for** loop only involves merge operations and there can be at most n merges, the desired result follows. □

3.3 Further Discussion

Algorithm 2 enjoys several nice features that Algorithm 1 and other relevant algorithms do not possess. For one thing, the initial partition J_0 of Algorithm 2 can be chosen arbitrarily. This allows the

algorithm to be warm-started if one has a good optimal partition estimate. This feature is particularly appealing when a sequence of related instances of (IR) need to be solved [10].

Another interesting feature of Algorithm 2 is its potential to be parallelized. This is possible since Algorithm 2 allows for multiple independent merge operations in each iteration; see Step 8 of Algorithm 2. As an illustrative example with $y = \{6, 4, 2, 9, 11, 4\}$ and $\omega = e$, we demonstrate in Figure 1 the different behavior between Algorithm 2 and Algorithm 1 when applied on this data set.

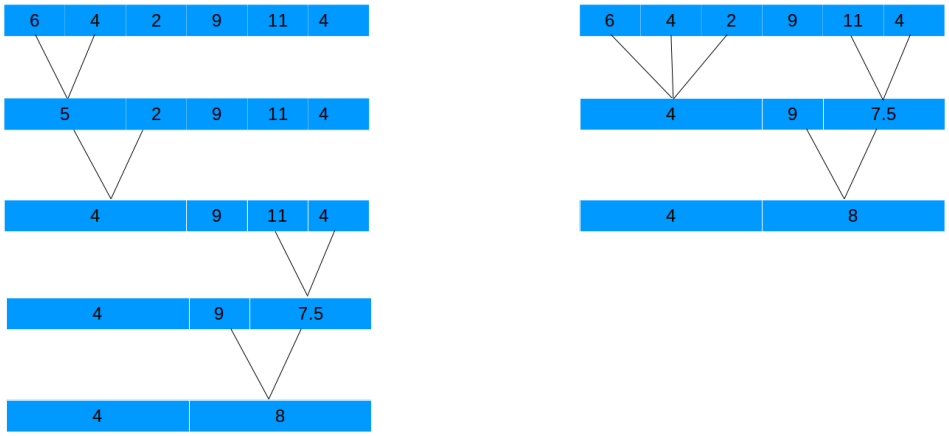


Figure 1: Illustration of per-iteration merge operations in PAV (left) and PDAS (right).

As illustrated in Figure 1, each iteration of PAV only merges two consecutive blocks whereas PDAS allows multiple consecutive blocks to be merged simultaneously throughout the dataset. Notice also that PDAS only requires three division operations while PAV requires four, despite the fact that in both methods the number of merge operations are counted as four.

4 PDAS for Trend Filtering

The trend filtering problem (TF) can be viewed as a generalization of problem (IR). While (IR) imposes monotonicity on the solution vector θ, variants of (TF) can impose other related properties, as illustrated in §4.1. Consequently, it is natural to extend PDAS for solving (TF), as we do in §4.2. However, since a direct application of a PDAS method may cycle when solving certain versions of (TF), we propose safeguarding strategies to ensure convergence; see §4.3.

4.1 Regularization with Difference Operators

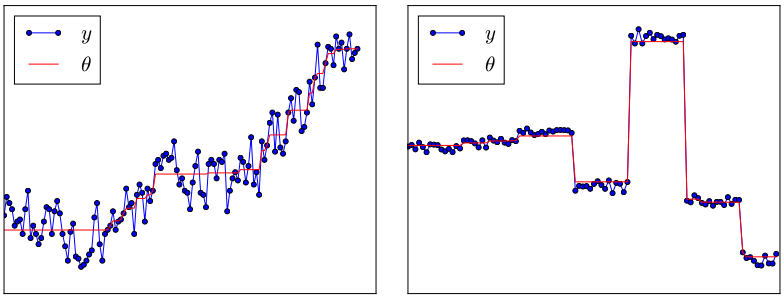
Common choices for the regularization function in problem (TF) are $g(\theta) = g_1(\theta) := \|D^{(d,n)}\theta\|_1$ or $g(\theta) = g_{1+}(\theta) = \|(D^{(d,n)}\theta)_+\|_1$, where $D^{(d,n)} \in \mathbb{R}^{(n-d)\times n}$ is the d-order difference matrix on $\mathbb{R}^n$. The choice of the regularization function determines the properties that one imposes on θ. We illustrate the typical behavior of θ for different choices of the regularization in Figure 2.

As shown in Figure 2, when $g = g_{1+}$ the fitted variable θ has the property of being nearly-monotone and nearly-convex for $d = 1$ and $d = 2$, respectively. Similarly, when $g = g_1$, the fitted curve would be piecewise constant and piecewise linear for $d = 1$ and $d = 2$, respectively. Higher order difference operators may be used, though the first and second order ones are more widely used.

4.2 A PDAS Framework for Trend Filtering

For brevity, let $D := D^{(d,n)}$. Denote the optimal solution of problem (TF) as (θ^*, z^*). Corresponding to this optimal solution, we may partition the indices of $D\theta^*$ as follows:

$$\mathcal{P}^* = \{j : (D\theta^*)_j > 0\};\ \ \mathcal{N}^* = \{j : (D\theta^*)_j < 0\};\ \ \mathcal{A}^* = \{j : (D\theta^*)_j = 0\}.$$



(a) $g = g_{1+}, d = 1$ (b) $g = g_1, d = 1$

(c) $g = g_{1+}, d = 2$ (d) $g = g_1, d = 2$

Figure 2: Trend filtering solutions for different choices of g and $D^{(d,n)}$.

A typical PDAS framework consists of three steps, as shown in Algorithm 3: subspace minimization (SSM), termination check, and partition update. We now discuss each of these steps in turn.

Algorithm 3 PDAS Framework

1: Input an initial partition $(\mathcal{P}, \mathcal{N}, \mathcal{A})$
2: **loop**
3: Compute the subspace minimizer (θ, z) corresponding to $(\mathcal{P}, \mathcal{N}, \mathcal{A})$ ▷ subspace minimization (SSM)
4: **if** (θ, z) is optimal, **then** terminate and return (θ, z) ▷ termination check
5: Compute a new partition $(\mathcal{P}, \mathcal{N}, \mathcal{A})$ ▷ partition update

Subspace Minimization Corresponding to an estimate $(\mathcal{P}, \mathcal{N}, \mathcal{A})$ of the optimal partition $(\mathcal{P}^*, \mathcal{N}^*, \mathcal{A}^*)$ there exists a unique primal-dual estimate (θ, z) of (θ^*, z^*). Denoting $\mathcal{I}$ as the union $\mathcal{P} \cup \mathcal{N}$, the following schematics show the processes for computing (θ, z) for problem (TF).

SSM for $g(\theta) = \|(D\theta)_+\|_1$
Set $z_j \leftarrow 0$ for $j \in \mathcal{N}$ and $z_j \leftarrow 1$ for $j \in \mathcal{P}$.
Solve for $(\theta, z_{\mathcal{A}})$:

$$\begin{bmatrix} I & \lambda D_{\mathcal{A}}^T \\ \lambda D_{\mathcal{A}} & 0 \end{bmatrix} \begin{bmatrix} \theta \\ z_{\mathcal{A}} \end{bmatrix} = \begin{bmatrix} y - \lambda D_{\mathcal{I}}^T z_{\mathcal{I}} \\ 0 \end{bmatrix}. \quad (1)$$

Set

$$\begin{aligned} \mathcal{V}_P &\leftarrow \{j \in \mathcal{P} : (D\theta)_j < 0\}; \\ \mathcal{V}_N &\leftarrow \{j \in \mathcal{N} : (D\theta)_j > 0\}; \\ \mathcal{V}_{AP} &\leftarrow \{j \in \mathcal{A} : z_j > 1\}; \\ \mathcal{V}_{AN} &\leftarrow \{j \in \mathcal{A} : z_j < 0\}. \end{aligned}$$

SSM for $g(\theta) = \|D\theta\|_1$
Set $z_j \leftarrow -1$ for $j \in \mathcal{N}$ and $z_j \leftarrow 1$ for $j \in \mathcal{P}$.
Solve for $(\theta, z_{\mathcal{A}})$:

$$\begin{bmatrix} I & \lambda D_{\mathcal{A}}^T \\ \lambda D_{\mathcal{A}} & 0 \end{bmatrix} \begin{bmatrix} \theta \\ z_{\mathcal{A}} \end{bmatrix} = \begin{bmatrix} y - \lambda D_{\mathcal{I}}^T z_{\mathcal{I}} \\ 0 \end{bmatrix}. \quad (2)$$

Set

$$\begin{aligned} \mathcal{V}_P &\leftarrow \{j \in \mathcal{P} : (D\theta)_j < 0\}; \\ \mathcal{V}_N &\leftarrow \{j \in \mathcal{N} : (D\theta)_j > 0\}; \\ \mathcal{V}_{AP} &\leftarrow \{j \in \mathcal{A} : z_j > 1\}; \\ \mathcal{V}_{AN} &\leftarrow \{j \in \mathcal{A} : z_j < -1\}. \end{aligned}$$

Note that the solution $(\theta, z_{\mathcal{A}})$ of system (1) or (2) could be efficiently obtained by

$$\text{solving for } z_{\mathcal{A}} \text{ the system} \qquad D_{\mathcal{A}} D_{\mathcal{A}}^T z_{\mathcal{A}} = D_{\mathcal{A}}(y - \lambda D_{\mathcal{I}}^T z_{\mathcal{I}})/\lambda, \tag{3a}$$

$$\text{then setting} \qquad \theta \leftarrow y - \lambda D^T z. \tag{3b}$$

When $D_{\mathcal{A}}$ is a first (second) order difference matrix, $D_{\mathcal{A}} D_{\mathcal{A}}^T$ is a tridiagonal (quindiagonal) matrix, meaning that the left-hand side match in (3a) is banded and $z_{\mathcal{A}}$ can be obtained cheaply.

Termination Check One can easily show that a computed pair (θ, z) is optimal if the set $\mathcal{V} = \mathcal{V}_P \cup \mathcal{V}_N \cup \mathcal{V}_{AP} \cup \mathcal{V}_{AN}$, consisting of indices of $D\theta$ and z corresponding to violated bounds, is empty [19]. Thus, when $\mathcal{V}$ is empty, optimality has been reached and the algorithm terminates. Otherwise, these sets indicate a manner in which the partition could be updated.

Partition Update In PDAS methods, the indices of $D\theta$ and z violating their bounds are deemed candidates for being switched from one index set in a partition to another. Specifically, a standard update in a PDAS method [17] involves the following steps:

$$\begin{aligned} &\mathcal{P} \leftarrow (\mathcal{P} \backslash \mathcal{V}_P) \cup \mathcal{V}_{AP}; \\ &\mathcal{N} \leftarrow (\mathcal{P} \backslash \mathcal{V}_N) \cup \mathcal{V}_{AN}; \\ &\mathcal{A} \leftarrow \mathcal{A} \backslash (\mathcal{V}_{AP} \cup \mathcal{V}_{AN}) \cup (\mathcal{V}_P \cup \mathcal{V}_N). \end{aligned} \tag{4}$$

4.3 Safeguarding

There is no convergence guarantee for Algorithm 3 for an arbitrary instance of (TF); indeed, an example illustrating that the method can cycle when solving a BQP is given in [18]. This example is presented here to illustrate that Algorithm 3 can cycle for certain instances of (TF).

Example 4.1. *Let* $y = (603, 996, 502, 19, 56, 139)^T$, $\lambda = 100$, *and* $g(\theta) = \|D^{(2,6)}\theta\|_1$. *The iterates produced in the first few iterations of Algorithm 3 are shown in Table 1. Since the algorithm returns*

Iter	$\mathcal{P}$	$\mathcal{N}$	$\mathcal{A}$	$D\theta$	z
0	$\{2,3,4\}$	$\{1\}$	$\emptyset$	$(13, -689, 820, -254)^T$	$(-1, 1, 1, 1)^T$
1	$\{3\}$	$\emptyset$	$\{1,2,4\}$	$(0, 0, \frac{4227}{38}, 0)^T$	$(-\frac{5293}{2280}, -\frac{482}{475}, 1, \frac{5201}{5700})^T$
2	$\{3\}$	$\{1,2\}$	$\{4\}$	$(-787, 520, -16, 0)^T$	$(-1, -1, 1, \frac{91}{100})^T$
3	$\emptyset$	$\{1\}$	$\{2,3,4\}$	$(-\frac{887}{5}, 0, 0, 0)^T$	$(-1, \frac{127}{125}, \frac{371}{125}, \frac{943}{500})^T$
4	$\{2,3,4\}$	$\{1\}$	$\emptyset$	$(13, -689, 820, -254)^T$	$(-1, 1, 1, 1)^T$
$\vdots$					

Table 1: An illustration of Algorithm 3 cycling

to a previously explored partition without computing an optimal solution, the algorithm cycles, i.e., it is not convergent for this problem instance from the given starting point.

A simple safeguarding strategy to overcome this issue and ensure convergence is proposed in [20] and subsequently embedded in the work of [21]. In particular, when $|\mathcal{V}|$ fails to decrease for several consecutive iterations, a backup procedure is invoked in which (4) is modified to only change partition membership of one index of $\mathcal{V}$. We present this approach in Algorithm 4.

The safeguard of Algorithm 4 employs a heuristic to decide whether to update the partition by (4) or (5). However, we have found an alternative strategy that performs better in our experiments. First, unlike that of [20, 21], our safeguard changes the memberships of a portion of $\mathcal{V}$, where the portion size is dynamically updated. Another difference in the safeguard design is that we employ a finite queue (first-in-first-out) to store recent values of $|\mathcal{V}|$ of which the maximum serves as the reference measure that diminishes as the algorithm continues. When an element is pushed into the queue that is full, the earliest element is removed. We present our approach in Algorithm 5.

Algorithm 5 can be understood as follows. If $|\mathcal{V}| = 0$, then (θ, z) is optimal and the algorithm terminates. Otherwise, the update (4) is to be applied using only the $\lfloor p|\mathcal{V}| \rfloor$ indices from $\mathcal{V}$ corresponding to the largest violations. If $p = 1/|\mathcal{V}|$, then this corresponds to moving only one index as in [20], but

Algorithm 4 PDAS Framework with Safeguarding [21]

Input $(\mathcal{P}, \mathcal{N}, \mathcal{A})$ and an integer $\mathtt{t}_{\mathtt{max}}$
Initialize $\mathtt{V}_{\mathtt{best}} \leftarrow \infty$ and $t \leftarrow 0$
loop
  Compute the subspace minimizer (θ, z) corresponding to $(\mathcal{P}, \mathcal{N}, \mathcal{A})$
  if $|\mathcal{V}| = 0$, **then** terminate and return (θ, z)
  if $|\mathcal{V}| < \mathtt{V}_{\mathtt{best}}$ **then**
    Set $t \leftarrow 0$ and $\mathtt{V}_{\mathtt{best}} \leftarrow |\mathcal{V}|$
    Apply (4)
  else if $|\mathcal{V}| \geq \mathtt{V}_{\mathtt{best}}$ **then**
    Set $t \leftarrow t + 1$
    if $t \leq \mathtt{t}_{\mathtt{max}}$ **then**
      Apply (4)
    else if $t > \mathtt{t}_{\mathtt{max}}$ **then**
      Set $j \leftarrow \min\{i : i \in \mathcal{V}\}$ and apply a partition update by

$$\begin{array}{l} \text{moving } j \text{ from } \mathcal{P} \text{ to } \mathcal{A}, \text{ if } j \in \mathcal{V}_P \\ \text{moving } j \text{ from } \mathcal{N} \text{ to } \mathcal{A}, \text{ if } j \in \mathcal{V}_N \\ \text{moving } j \text{ from } \mathcal{A} \text{ to } \mathcal{P}, \text{ if } j \in \mathcal{V}_{AP} \\ \text{moving } j \text{ from } \mathcal{A} \text{ to } \mathcal{N}, \text{ if } j \in \mathcal{V}_{AN} \end{array} \tag{5}$$

Algorithm 5 PDAS Framework with Safeguarding

Input $(\mathcal{P}, \mathcal{N}, \mathcal{A})$, queue Q_m with size m, proportion $p \in (0, 1]$, parameter $\delta_s \in (0, 1)$ and $\delta_e \in (1, \infty)$
loop
  Compute the subspace minimizer (θ, z) corresponding to $(\mathcal{P}, \mathcal{N}, \mathcal{A})$
  if $|\mathcal{V}| = 0$, **then** terminate and return (θ, z)
  Set $\mathtt{max}/\mathtt{min} \leftarrow$ maximum/minimum of Q_m
  if $|\mathcal{V}| > \mathtt{max}$ **then**
    set $p \leftarrow \max(\delta_s p, \frac{1}{|\mathcal{V}|})$
  else if $|\mathcal{V}| < \mathtt{min}$ **then**
    push $|\mathcal{V}|$ into Q_m and set $p \leftarrow \max(\delta_e p, 1)$
  else
    push $|\mathcal{V}|$ into Q_m
  Sort $\mathcal{V}$ by $\max(\lambda|D\theta|, |z|)$ and apply (4), only changing the top $p|\mathcal{V}|$ indices

if $p \in (1/|\mathcal{V}|, 1]$, then a higher portion of violated indices may be moved. As long as the reference value—i.e., the maximum of the values in the queue—decreases, the value for p is maintained or is increased. However, if the reference value fails to decrease, then p is decreased. Overall, since the procedure guarantees that the reference value is monotonically decreasing and that p is sufficiently reduced whenever a new value for $|\mathcal{V}|$ is not below the reference value, our strategy preserves the convergence guarantees established in [20] while yielding better results in our experiments.

5 Experiments

We implemented Algorithms 2, 3, 4, and 5 in Python 2.7, using the Numpy (version 1.8.2) and Scipy (version 0.14.0) packages for matrix operations. In the following subsections, we discuss the results of numerical experiments for solving randomly generated instances of problems (IR) and (TF). For problem (IR), merge operations for Algorithm 2 were carried out sequentially (but recall Figure 1 in which we have illustrated that they could be carried out in parallel). Throughout our experiments, we set ω as an all-one vector.

5.1 Test on Isotonic Regression

We compare the performance of Algorithm 2 (`PDAS`) with a Python implementation of Algorithm 1 (`PAV`) that was later integrated into scikit-learn (version 0.13.1) [22]. The data y_i are generated by $y_i \leftarrow i + \varepsilon_i$ where $\varepsilon_i \sim \mathcal{N}(0, 4)$. The initial partition is set as $J_0 = \{\{1\}, \{2\}, \ldots, \{n\}\}$ for both algorithms. We generated 10 random instances each for $n \in \{1 \times 10^4, 5 \times 10^4, \ldots, 33 \times 10^4\}$.

Boxplots for running time in seconds (`Time (s)`) and number of merge operations (`# Merge`) are reported in Figure 3.

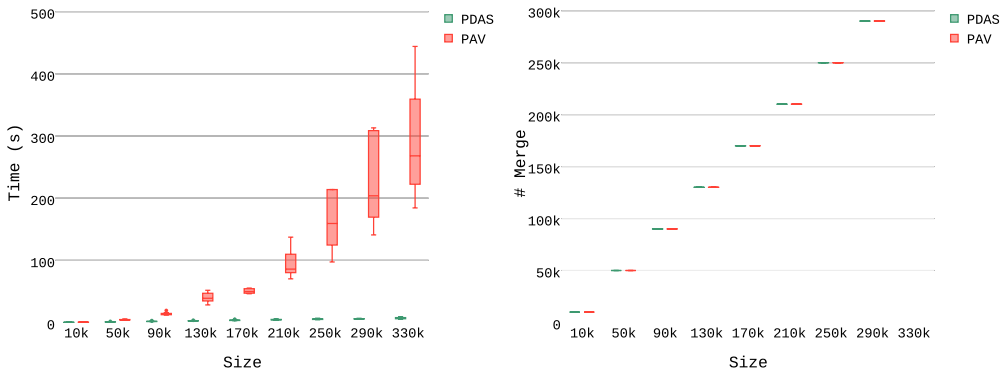


Figure 3: Comparison of PDAS and PAV in running time (left) and # of merges (right).

Figure 3 demonstrates that our implementation of `PDAS` outperforms `PAV` in terms of running time. That being said, when both use the set of singletons as the starting point, the numbers of merge operations performed by the two algorithms are nearly identical.

Warm-starting Figure 3 does not show an obvious advantage of `PDAS` over `PAV` in terms of the numbers of merge operations. However, as claimed, we now show an advantage of `PDAS` in terms of its ability to exploit a good initial partition. We simulate warm-starting `PDAS` by generating an instance of (IR) as in our previous experiment, solving it with `PDAS`, and using the solution as the starting point for solving related instances for which the data vector y has been perturbed. In particular, for each problem size n, we generated 10 perturbed instances by adding a random variable $\epsilon_i \sim \mathcal{N}(0, 10^{-2})$ to each y_i. The results of solving these instances are reported in Figures 4 and 5.

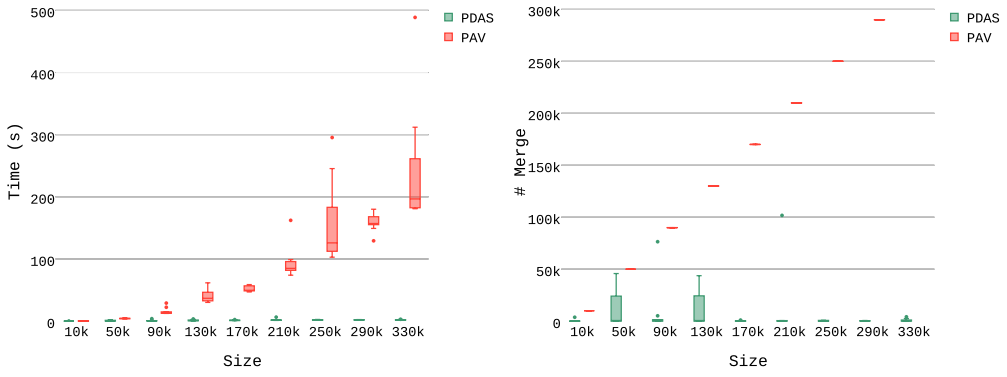


Figure 4: Comparison of warm-started PDAS and PAV in running time (left) and # of merges (right).

Since `PAV` is not able to utilize a good initial partition, the required work (i.e., number of merge operations) for solving the perturbed problems is not cheaper than for the base instance. In contrast, warm-starting the `PDAS` algorithm can greatly reduce the computational cost as observed in the much-reduced number of merge operations (even after accounting for the added split operations).

5.2 Test on Trend Filtering

We now compare the performance of several PDAS variants for trend filtering. In particular, we compare the straightforward PDAS method of Algorithm 3 (`PDAS`), Algorithm 4 (`SF1`), and the PDAS method with our proposed safeguard strategy in Algorithm 5 (`SF2`). The safeguard parameter

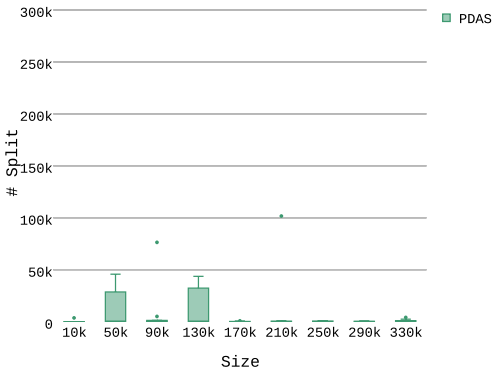


Figure 5: Summary of warm-started PDAS in terms of # of splits.

$\texttt{t}_{\texttt{max}} = 5$ was chosen for SF1 and we similarly set $m = 5$, $\delta_s = 0.9$, and $\delta_e = 1.1$ for SF2. We generated 10 random problem instances each for $n \in \{10^4, 1.7 \times 10^5, 3.3 \times 10^5\}$ where, for each instance, the data vector had y_i uniformly distributed in $[0, 10]$. Such datasets had minimum pattern and thus made each problem relatively difficult to solve. We considered both regularization functions g_1 and g_{1+} defined in §4.1 with difference matrices $D^{(1,n)}$ and $D^{(2,n)}$, setting $\lambda = 10$ in all cases. For all runs, we set an iteration limit of 800; if an algorithm failed to produce the optimal solution within this limit, then the run was considered a failure. The percentages of successful runs for each algorithm is reported in Table 2.

Table 2: Percentages of successful runs for each algorithm and problem type

n (size)	% of success											
	$g(\theta) = \|(D^{(1,n)}\theta)_+\|_1$			$g(\theta) = \|D^{(1,n)}\theta\|_1$			$g(\theta) = \|(D^{(2,n)}\theta)_+\|_1$			$g(\theta) = \|D^{(2,n)}\theta\|_1$		
	PDAS	SF1	SF2	PDAS	SF1	SF2	PDAS	SF1	SF2	PDAS	SF1	SF2
1.0e+4	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0
1.7e+5	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
3.3e+5	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0

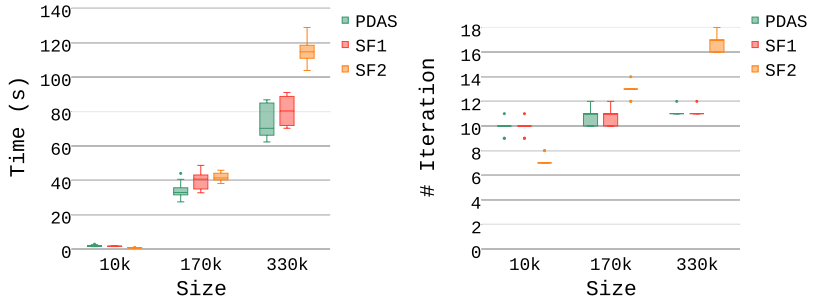
We observe from Table 2 that all algorithms solved all instances when the regularization function involved a first-order difference matrix, but that PDAS and SF1 both had failures when a second-order difference matrix is used. By contrast, our proposed safeguard in SF2 results in a method that is able to solve all instances within the iteration limit. This shows that our proposed safeguard, which allows more aggressive updates, can be more effective than a conservative safeguard.

To compare further the performance of the algorithms, we collected the running time and iteration number for all successfully solved instances. Figure 6 demonstrates that when $D = D^{(1,n)}$, all algorithms show very similar performance. However, when $D = D^{(2,n)}$ as in Figure 7, the results show that SF2 is not only more reliable than PDAS and SF1; it is also more efficient even when SF1 is successful. We also include the results for the interior-point method (IPM) proposed in [11], but emphasize that this algorithm is implemented in Matlab (as opposed to Python) and is only set up to solve the instances when an ℓ_1-regularization function is used. Although the interior point method demonstrates impressive performance, we remark that in general it is difficult to warm-starting interior point methods [23], despite recent efforts toward this direction [24, 25, 26] .

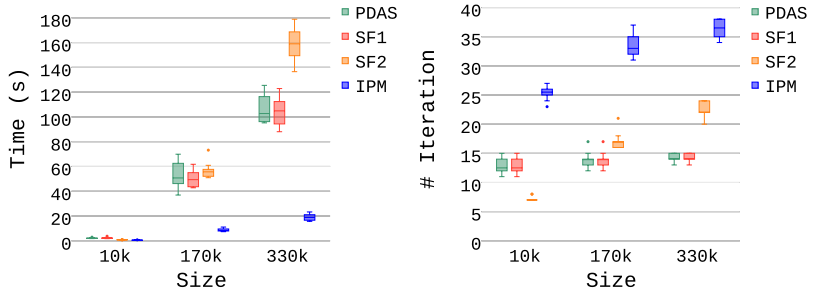
Warm-starting

We conclude our experiments by comparing the performance of IPM—which is not set up for warm-starting—and warm-started SF2. As in §5.1, we generated 10 perturbed instances for a given dataset by adding a random variable $\epsilon_i \sim \mathcal{N}(0, 10^{-2})$ to y_i. The running times and numbers of iterations for solving the perturbed problems are reported via boxplots in Figure 8.

Comparing the performance of SF2 between Figures 6, 7, and 8 shows that warm-starting SF2 can dramatically reduce the cost of solving an instance of (TF). With a cold-start, SF2 may require

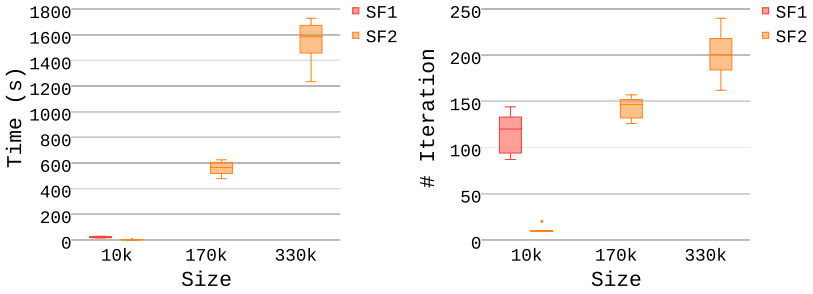


(a) $D = D^{(1,n)}$, $g = g_{1+}$

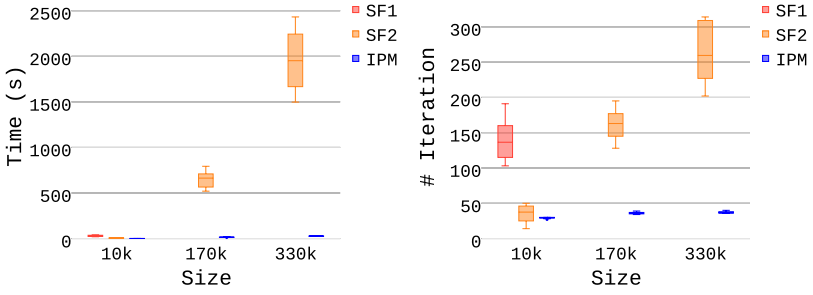


(b) $D = D^{(1,n)}$, $g = g_1$

Figure 6: PDAS vs IPM for $D = D^{1,n}$ and different choices of g.



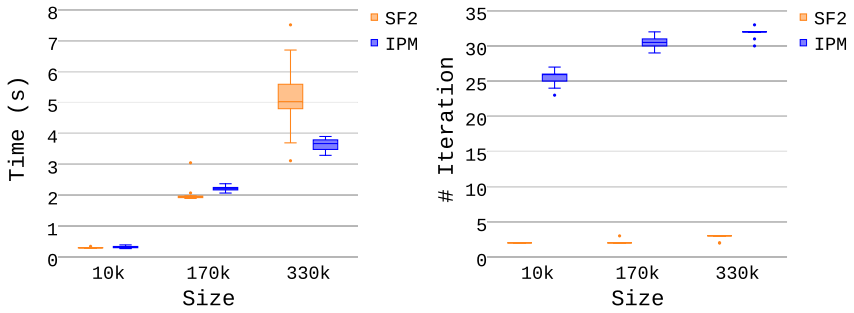
(a) $D = D^{(2,n)}$, $g = g_{1+}$



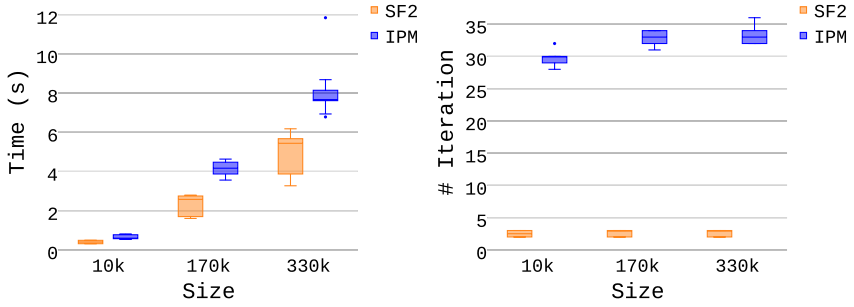
(b) $D = D^{(2,n)}$, $g = g_1$

Figure 7: PDAS vs IPM for $D = D^{2,n}$ and different choices of g.

hundreds of iterations, while with warm-starting it requires dramatically fewer iterations. In contrast, `IPM` does not benefit much from a good starting point.



(a) $D = D^{(1,n)}$, $g = g_1$



(b) $D = D^{(2,n)}$, $g = g_1$

Figure 8: PDAS (with warm-start) vs IPM.

6 Concluding Remarks

We propose innovative PDAS algorithms for Isotonic Regression (IR) and Trend Filtering (TF). For IR, our PDAS method enjoys the same theoretical properties as the well-known PAV method, but also has the ability to be warm-started, can exploit parallelism, and outperforms PAV in our experiments. Our proposed safeguarding strategy for a PDAS method for TF also exhibits reliable and efficient behavior. Overall, our proposed methods show that PDAS frameworks are powerful when solving a broad class of regularization problems.

References

[1] R. E. Barlow and H. D. Brunk. The isotonic regression problem and its dual. *Journal of the American Statistical Association*, 67(337):140–147, 1972.

[2] R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk. *Statistical Inference under Order Restrictions: Theory and Application of Isotonic Regression*. John Wiley & Sons Ltd., 1972.

[3] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-Scale Bayesian Click-Through rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine. In Johannes Frnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 13–20. Omnipress, 2010.

[4] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica. Ad Click Prediction: A View from the Trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1222–1230, New York, NY, USA, 2013. ACM.

[5] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 625–632, New York, NY, USA, 2005. ACM.

[6] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 694–699, New York, NY, USA, 2002. ACM.

[7] M. J. Best and N. Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1-3):425–439, 1990.

[8] A. J. Kearsley, R. A. Tapia, and M. W. Trosset. An approach to parallelizing isotonic regression. In *Applied Mathematics and Parallel Computing*, pages 141–147. Springer, 1996.

[9] M. Zapletal. Isotonic regression implementation in apache spark. `http://www.cakesolutions.net/teamblogs/isotonic-regression-implementation-in-apache-spark`, 2015.

[10] X. Suo and R. Tibshirani. An ordered lasso and sparse time-lagged regression. *Technometrics*, DOI: 10.1080/00401706.2015.1079245, 2015.

[11] S. Kim, K. Koh, S. Boyd, and D. Gorinevsky. ℓ_1 trend filtering. *SIAM Review*, 51(2):339–360, 2009.

[12] A. Ramdas and R. J. Tibshirani. Fast and Flexible ADMM Algorithms for Trend Filtering. *Journal of Computational and Graphical Statistics*, DOI: 10.1080/10618600.2015.1054033, 2014.

[13] Á. Barbero and S. Sra. Modular proximal optimization for multidimensional total-variation regularization. *ArXiv e-prints, 1411.0589*, 2014.

[14] R. J. Tibshirani, H. Hoefling, and R. Tibshirani. Nearly-isotonic regression. *Technometrics*, 53(1):54–61, 2011.

[15] S. J. Grotzinger and C. Witzgall. Projections onto order simplexes. *Applied Mathematics and Optimization*, 12(1):247–270, 1984.

[16] M. Aganagić. Newton's method for linear complementarity problems. *Mathematical Programming*, 28(3):349–362, 1984.

[17] M. Hintermüller, K. Ito, and K. Kunisch. The primal-dual active set strategy as a semismooth Newton method. *SIAM Journal on Optimization*, 13(3):865–888, 2003.

[18] F. E. Curtis, Z. Han, and D. P. Robinson. A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization. *Computational Optimization and Applications*, 60(2):311–341, 2014.

[19] F. E. Curtis and Z. Han. Globally Convergent Primal-Dual Active-Set Methods with Inexact Subproblem Solves. Technical Report 14T-010, COR@L Laboratory, Department of ISE, Lehigh University, 2014.

[20] J. Kim and H. Park. Fast active-set-type algorithms for l1-regularized linear regression. In *International Conference on Artificial Intelligence and Statistics*, pages 397–404, 2010.

[21] R. H. Byrd, G. M. Chin, J. Nocedal, and F. Oztoprak. A family of second-order methods for convex ℓ_1-regularized optimization. Technical report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA, 2012.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] E. John and E. A. Yıldırım. Implementation of warm-start strategies ininterior-point methods for linear programming infixed dimension. *Computational Optimization and Applications*, 41(2):151–183, 2007.

[24] E. A. Yıldırım and S. J. Wright. Warm-start strategies in interior-point methods for linear programming. *SIAM Journal on Optimization*, 12(3):782–810, 2002.

[25] J. Gondzio and T. Terlaky. A computational view of interior point methods. In J. E. Beasley, editor, *Advances in Linear and Integer Programming*, pages 103–144. Oxford University Press, Inc., New York, NY, USA, 1996.

[26] J. Gondzio and A. Grothey. Reoptimization with the primal-dual interior point method. *SIAM Journal on Optimization*, 13(3):842–864, 2002.